一、基础选择(每题 2 分，共 30 分)

1. （　　）的出现，推动了个人计算机的诞生。

   A. 晶体管　　B. 电子管　　C. 大规模集成电路　　D. 集成电路

2. 一块大小为 5*5 的空地，在其中选 5 块，使得任意两块不在同一行同一列的方案数量是（　　）

   A. 120　　B. 24　　C. 720　　D. 60

3. 在 8 位二进制补码中，10101010 表示的数是十进制下的：（　）

   A.176　　B. -86　　C. -85　　D. -84

4. 关键字序列（8，9，10，4，5，6，20，1，2）只能是下列排序算法中(　)的两趟排序后的结果。

   A．选择排序　　B.冒泡排序　　C.插入排序　　D.快速排序

5. 在含 n 个顶点和 e 条边的无向图的邻接矩阵中，零元素的个数为（　　）

   A. $(n-1)^2-e$　　B. $2e$　　C. n^2-e　　D. n^2-2e

6. 二维数组 A 的每个元素是由 10 个字符组成的串，其行下标 $i=0,1,\ldots,8$，列下标 $j=1,2,\ldots,10$。若 A 按行先存储，元素 `A[8][5]` 的起始地址与当 A 按列先存储时的元素（　）的起始地址相同。设每个字符占一个字节。

   A．`A[8][5]`　　B．`A[3][10]`　　C．`A[5][8]`　　D．`A[0][9]`

7. 若一棵二叉树具有 10 个度为 2 的结点，5 个度为 1 的结点，则度为 0 的结点的个数是(　)。

   A.9　　B.11　　C.15　　D.不能确定

8. 某文本包含 240 个汉字、39 个数字以及 666 个字母，若将其强制转换为一个 char 数组，则数组的长度为（　）

   A.945　　B.279　　C.1851　　D.1185

9. 前序遍历序列与中序遍历序列相同的二叉树为（　）

   A.根结点无左子树的二叉树

   B.根结点无右子树的二叉树

   C.只有根结点的二叉树或非叶子结点只有左子树的二叉树

   D.只有根结点的二叉树或非叶子结点只有右子树的二叉树

10. 下列说法正确的是（　） 算法

    A. SPFA算法无法用来判断给定图是否存在负环

    B. 当图中不存在负权环但是存在负权边，Dijkstra算法一定不能求最短路

    C. 当图中不存在负权环但是存在负权边，bellman-ford算法一定能求最短路

    D. 相比于稀疏图，在稠密图上更适合使用SPFA算法

组合人送大家的填空题

11. 一枚质地均匀骰子，六个面分别是 1 到 6，骰子投三次，三次数值的加和不超过5的概率是（　）

12. A、B、C、D、E五人吃饭，围成圆桌，要求A和D不相邻，方案数有 （　　）种

13. 小明和小红石头剪刀布，小明会各以 1/2 的概率猜石头和剪刀，小红会各以 1/3 的概率猜石头剪刀布，小明单局游戏的胜利概率是?

14. 现在有5个有标号的盒子，你有10个完全一样的小球，你可以将小球随意地放置到盒子中，在所有的放置方案中，使得至少有一个盒子为空的方案数占总方案数中的比例是（　　）

15. 使三角形三边之和为8，且要求三角形三条边都是正整数，在所有的构成方案中，三角形是锐角三角形的概率是（　　）

二、程序阅读(判断题每题 1.5 分，选择题每题 3 分，共 40 分)

程序阅读第一题(共 13 分)

本题输入仅一行，由一个仅包含大写字母的字符串和一个正整数组成。

```
01  #include <iostream>
02  using namespace std;
03  int cnt[30], k, r, ans;
04  int main(){
05      string s;
06      cin >> s >> k;
07      cnt[s[r] - 'A']++;
08      for(int i = 0; i < s.size(); i++){
09          while(r < s.size() && r - i + 1 - cnt[s[i]-'A'] <= k){
10              r++;
11              if(r == s.size())   break;
12              cnt[s[r]-'A']++;
13          }
14          ans = max(ans, r - i);
15          cnt[s[i]-'A']--;
16      }
17      cout << ans << endl;
18  }
```

判断题

1. 删去 11 行，程序运行结果不会改变。
2. （2 分）删去 09 行 while 循环中的第一个条件 `r < s.size()`，程序运行结果不会改变。
3. 08 行的 for 循环中 `i` 的定义改为 `int i = 1`，程序运行结果不会改变。

选择题

1. 若字符串的长度为 5，k 的值为 1，则输出的值的最小可能是（　　）。

   A. 0　　B. 1　　C. 2　　D. 4

2. 若输入为 ABCBABAD 2，则输出为（　　）。

   A. 5　　B. 4　　C. 3　　D. 6

3. 假设读入的字符串长度为 n，读入的正整数为 k，则程序的时间复杂度为（　　）

   A. $O(n)$　　B. $O(n^2)$　　C. $O(n+k)$　　D. $O(nk)$

程序阅读第二题(共 12 分)

```
01  #include <bits/stdc++.h>
02  using namespace std;
03  int n;
04  int dfs(vector<int> a, int l = 0, int r = n - 1){
05      if(l == r && l != n / 2)    return 0;
06      int t = a[rand() % (r - l + 1)];
07      vector<int>low, up;
08      for(int i = 0; i < r - l + 1; i++){
09          if(a[i] < t){
10              low.push_back(a[i]);
11          }else if(a[i] > t){
```

```
            up.push_back(a[i]);
        }
    }
    if(l + (int)low.size() - 1 >= n / 2)
        return dfs(low, l, l + low.size() - 1);
    else if(r - up.size() + 1 <= n / 2)
        return dfs(up, l + low.size() + 1, r);
    else
        return t;
}
int main(){
    cin >> n;
    vector<int>a(n);
    for(int i = 0; i < n; i++){
        cin >> a[i];
    }
    cout << dfs(a);
}
```

判断题

1. 若将第六行改为 `int t = a[0]`，程序运行结果不会发生改变。
2. 若将第四行的后两个参数改为 `int l, int r`，程序运行结果不会发生改变。
3. 第五行的后面应该再加一个判断语句 `if(l > r) return 0;`，若不加，则程序可能会死循环。
4. 若输入的 n 的值为 1，则无论输入什么，输出都为 0。

选择题

1. 该程序的时间复杂度为（　　）

   A. $O(nlogn)$　　B. $O(n^2)$　　C. $O(logn)$　　D. $O(n)$

2. 若输入为 `6 1 10 7 8 2 2`，则输出为（　　）

   A. 6　　B. 7　　C. 2　　D. 8

程序阅读第三题（共 15 分）

本题是 NE 写的一个 hash_map，用于存储一些映射关系。

```
#include <bits/stdc++.h>
using namespace std;
#define ll long long
const int maxm = 1e6 + 5;
class hash_map {
public:
    struct node {
        ll u;
        ll v,next;
    } e[maxm<<1];
    ll head[maxm],nume,numk,id[maxm];
    ll& operator[](ll u) {
        int hs = u % maxm;
        for(int i = head[hs]; i; i = e[i].next)
            if(e[i].u == u) return e[i].v;
        if(!head[hs])   id[++numk] = hs;
        return e[++nume] = (node) {u,0,head[hs]}, head[hs] = nume,
e[nume].v;
    }
```

```
    void clear() {
        for(int i = 0; i <= numk; i++)
            head[id[i]] = 0;
        numk = nume = 0;
    }
} m;
int main() {
    int n;
    ll a, b, c;
    cin >> n;
    while(n--) {
        cin >> a;
        if(a == 1) {
            cin >> b >> c;
            m[b] = c;
        } else {
            cin >> b;
            cout << m[b] << endl;
        }
    }
}
```

判断题

1. 该 hash_map 只能存储整数与整数的对应关系，无法存储字符串或小数与整数的对应关系。
2. 首先输入 `1` 使得 `n=1`，然后再输入 `2 3`，输出为 0。
3. 若输入的所有 a 都为 1，则程序没有输出。
4. 若先输入 `3`，然后输入 `1 2 3\n 1 2 4\n 2 2` (其中 \n 为换行符)，则输出为 4。

选择题

1. 该代码解决哈希冲突的方式为?

   A. 链地址法　　B. 线性探测再散列法　　C. 再哈希法　　D. 该代码不解决冲突，只避免了冲突的发生。

2. 若已插入元素数量为 n，数组大小为 m（本代码中 m 为 10^6），则调用 clear() 函数的时间复杂度为?

   A. $O(n)$　　B. $O(m)$　　C. $O(n+m)$　　D. $O(nm)$

3. 若已插入元素数量为 n，且 n 的大小约为 $\frac{m}{100}$，则单次调用 `[]` 的平均复杂度和最坏复杂度为?

   A. $O(n), O(n)$　　B. $O(1), O(n)$　　C. $O(1), O(1)$　　D. $O(1), O(logn)$

三、程序填空(每题 3 分，共 30 分)

1.（树的重心）给出一个无根树，求树的重心。树的重心的定义：删除这个点及这条边相连的边之后，形成的所有连通块中最大连通块最小。若有多个符合条件的点，输出编号最小的那个。

思路：对每一个点的每一个子树求最大值，找到最大子树最小的点即可。

```
#include <bits/stdc++.h>
using namespace std;
const int maxn = 2e4 + 5;
int son[maxn], n, ans=1, size=1e5;
vector<int>G[maxn];
void dfs(int now, int pre) {
    int tmp = 0;
    son[now] = 1;
```

```
    for(int i = 0; i < G[now].size(); i++) {
        int nxt = G[now][i];
        if(nxt != pre) {
            dfs(nxt, now);
            son[now] += **1**;
            tmp = max(tmp, **2**);
        }
    }
    tmp = max(tmp, **3**);
    if(**4**){
        ans = now;
        size = tmp;
     }
}
int main() {
    cin >> n;
    for(int i = 1; i < n; i++) {
        int u, v;
        cin >> u >> v;
        G[u].push_back(v);
        G[v].push_back(u);
    }
    dfs(1,-1);
    cout << ans << endl;
}
```

1. 1 处应填

   A. son[nxt]　　B. son[nxt] + 1　　C. 1　　D. son[now]

2. 2 处应填

   A. son[now]　　B. son[now] + 1　　C. son[nxt] + 1　　D. son[nxt]

3. 3 处应填

   A. son[now]　　B. n-son[now]　　C. n-son[now]-1　　D. son[now]+1

4. 4 处应填

   A. tmp < size

   B. tmp <= size

   C. tmp * maxn + now < size * maxn + ans

   D. now * maxn + tmp < ans * maxn + size

2.（数字游戏）Alice 和 Bob 正在玩数字游戏。初始时两个人手中的数字分别为 a, b，两个人轮流操作，每次操作要么将手中的数字加上对方手中的数字，然后对 n 求余；要么用手中的数字乘以对方手中的数字，然后对 n 求余。得到的新数字替换自己手上原来的数字。假设 $n = 5, a = 2, b = 3$，那么 Alice 可以使用第一种操作使得自己手上的数字变成 0($(2 + 3)\%5 = 0$)，或者用第二种操作让自己手上的数字变成 1。

谁先让自己手上的数字变为 0 谁就获胜。输出 1 表示先手必胜，0 表示平局，-1 表示后手必胜。
$(1 \leq a, b < n \leq 1000)$

```
#include <bits/stdc++.h>
using namespace std;
vector<int> g[1100000];
int f[1100000];
int n, k, a, b;
```

```cpp
#define id(a,b) (a*n+b)
int cnt[1100000];
void init(int n) {
    for(int i = 1; i < n; i++)
        for(int j = 1; j < n; j++)
            g[**1**].push_back(id(i,j));
    for(int i = 1; i < n; i++)
        for(int j = 1; j < n; j++)
            g[id(j,(i+j)%n)].push_back(id(i,j));
}
int main() {
    cin >> n;
    init(n);
    queue<pair<int, int> > Q;
    for(int i = 0; i <= n * n; i++)
        cnt[i] = **2**;
    for(int i = 1; i < n; i++)
        Q.push(**3**);
    while(!Q.empty()) {
        pair<int, int> x = Q.front();
        Q.pop();
        for(int y : g[x.first]) {
            if (**4**)  continue;
            if (!x.second) {
                f[y] = 1;
                Q.push({y, 1});
            } else {
                cnt[y]--;
                if(!cnt[y]) {
                    f[y] = -1;
                    Q.push(**5**);
                }
            }
        }
    }
    cin >> a >> b;
    cout << f[**6**] << endl;
}
```

1. 1 处应填

   A. `i+j%n`　　B. `i+j`　　C. `i*j`　　D. `(i+j)%n`

2. 2 处应填

   A. 2　　B. 1　　C. n　　D. n/2

3. 3 处应填

   A. `{i,0}`　　B. `{i,1}`　　C. `{id(n*i), 0}`　　D. `{id(n*i), 1}`

4. 4 处应填

   A. `f[y] == 0`　　B. `f[y] != 0`　　C. `f[x.second] != 0`　　D. `f[x.second] == 0`

5. 5 处应填

   A. `{y,0}`　　B. `{y,-1}`　　C. `{y,1}`　　D. `{y, n}`

6. 6 处应填

   A. `a+b`　　B. `a*b%n`　　C. `(a+b)%n`　　D. `id(a,b)`